AP-710

APPLICATION NOTE

Migrating from the MCS®51 Microcontroller to the MCS 251 Microcontroller (8XC251SB) - Software and hardware considerations

SHU SHEN DANG
TECHNICAL MARKETING
EIGHT BIT MICROCONTROLLERS

February 1995

Order Number: 272672

Additional copies of this document or other Intel literature may be obtained from:

Intel Corporation
Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

 CG-041493

TABLE OF CONTENTS

1.0 INTRODUCTION

The Intel MCS ® 51 microcontroller is one of the most widely used 8-bit microcontroller in the world. Many applications today require a high performance 8-bit microcontrollers with more addressing space, register-based instructions, larger internal data RAM, more stack space and effective high level language programming. The next generation Intel MCS 251 microcontroller is designed to fulfill the high performance applications requirements and provide a natural upgrade path for the MCS 51 microcontroller.

The first product from the MCS 251 microcontroller family is the 8XC251SB. The 8XC251SB is both binary code and pin compatible with the 8XC51FX. Therefore the 8XC251SB can be plugged-into an 8XC51FX socket and the application will receive an immediate performance boost. The 8XC251SB has the same features as the 8XC51FX. These features include

- Three 16-bit Timers/Counters (Timer 0, 1 and 2)
- Programmable Serial Channel with
    - Framing Error Detection
    - Automatic Address Recognition
- Programmable Counter Array (PCA) with
    - High Speed Output
    - Real Time Capture/Compare
    - Pulse Width Modulation (PWM)
    - Software Watchdog Timer
- 32 Programmable I/O Lines
- 7 Interrupt Sources
- 16K On-Chip Program Memory
- Power Saving Idle and Power Down Modes
- ONCE (On-Circuit Emulation) Mode

The additional features which are also available in the 8XC251SB are:-

- Hardware Watchdog Timer
- 1K Bytes of On-Chip Data RAM
- Enriched instruction set with 16 bit and 32 bit capability
- Additional 8-bit, 16-bit and 32-bit registers with accumulator and index register functionality
- Configurable to 128K extended addressability
- Speed-up external code fetching in the Page Mode
- TRAP instruction

The 8XC251SB provides an easy, low cost, low risk and yet higher performance path for the MCS 51 microcontrollers. The block diagram of the 8XC251SB microcontroller is shown in Figure 1.

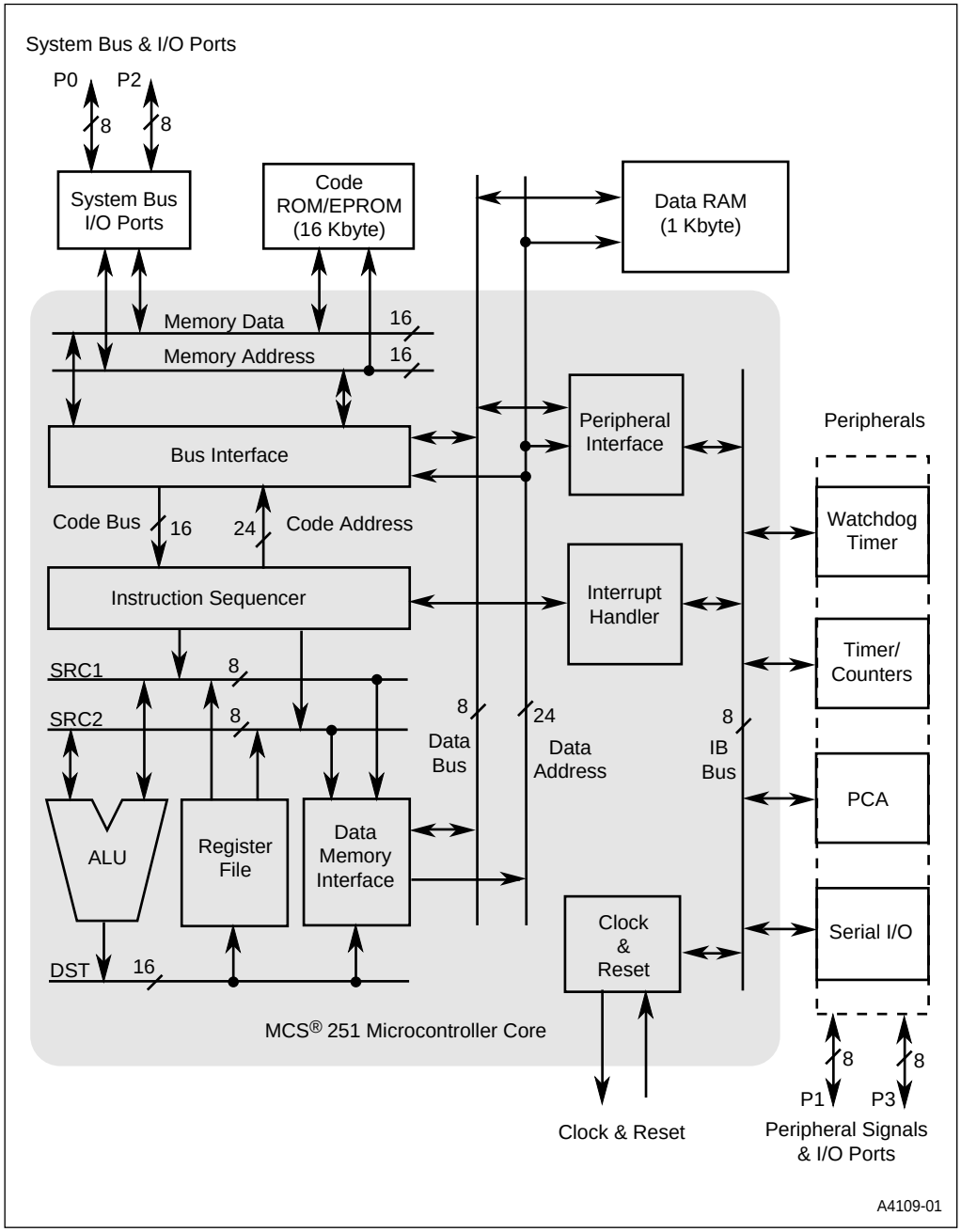

Figure 1. The 8XC251SB Microcontroller Block Diagram

This application note assumes that readers are familiar with the MCS ® 51 microcontrollers. The purpose of this application note is to address the software and hardware design considerations when migrating from the MCS 51 microcontroller to the first MCS 251 microcontroller, 8XC251SB. The focus is on the plug-in compatibility with the 8XC51FX. To plug-in the 8XC251SB into an 8XC51FX socket, the user must consider the programming of the configuration bytes that is explained in the next section.

The detailed 8XC251SB information can be obtained from the 8XC251SB User's Manual, Data sheet and applications notes that are listed in Section 12.0, Additional References.

2.0 USING CONFIGURATION BYTES

The 8XC251SB provides a variety of features and operating modes by programming two configuration bytes. The two configuration bytes are assigned with byte address 80H (Config0) and 81H (Config1) at TROM location which is a separate memory array. It is different from the User ROM location. By programming the configuration bytes, the 8XC251SB can be configured into the following modes:-

- Binary or Source Mode
- 0 or 1 Wait State generation
- ALE extension
- 2 or 4 bytes interrupt frame
- Extended addressability of 128K external memory
- Page Mode
- Mapping the internal EPROM for Look-up Table

These configuration bytes are the same for the OTP (87C251SB), Rom (83C251SB) and ROMless (80C251SB) parts. Even the ROMless part (80C251SB), it must be configured first before using it.

In the Binary Mode, the 8XC251SB is able to run 8XC51FX codes without recompilation. The full power of the 8XC251SB new instruction is utilized in the Source Mode. The detail explanation of these two modes is discussed in the Section 3.0, Code Compatibility.

The 8XC251SB can be used to interface with different speed of external devices. The 0 Wait State is used to interface with fast external devices (including memory) and 1 Wait State is used to interface with slow memory and external peripherals. For slower external devices, ALE is extended for an additional state as well. More discussion about the Wait State is explained in the Section 4.0, Wait State Generation.

For the 8XC51FX compatibility, the 8XC251SB is configured to 2 bytes frame pushed on the stack during call to Interrupt Service Routines (ISR). The other option is to have 4 bytes frame pushed on the stack during call to Interrupt Service Routine. The four bytes consist of 3 bytes of the Program Counter (PC) and the PSW1 byte. Further explanation on this is discussed in the Section 5.0, Extended Stack.

The 8XC251SB can be configured to have extended addressability with 128K for external memory devices. The detailed information about this configuration is explained in the Section 10.0, Extended Memory Space.

In the Page Mode configuration, the 8XC251SB reduces the time required for external code fetching cycles to half. More discussion about the page mode design is explained in the Section 11.0, Page Mode Design.

One of the configuration bits of the 8XC251SB is used to map the upper 8K bytes of internal EPROM to data memory addresses 00:E000H-00:FFFFH as Look-up Table.

Table 1 and Table 3 show the details of the bits assignment for the configuration bytes Config0 and Config1 respectively. The bits that are not defined are reserved for future use. The value found in the reserved bits should be ignored and not to be changed. Table 2 explains the configuration bit setting in Config0 that control the extended addressing capability of the 8XC251SB.

Table 1. CONFIG0: Configuration Bytes

CONFIG0 TROM Address = 80H Reset value = 11111111B

	-	-	WSa	XALE	Rd1	Rd0	Page	Src
bit	7	6	5	4	3	2	1	0

Symbol	Function
Src	Configures in Source or Binary Mode = 1 is a Source code compatible mode which fully utilizes the power of the new 8XC251SB instructions, some old MCS ® 51 instructions require A5 prefix = 0 is a Binary mode which is binary code compatible with MCS 51 microcontrollers, new 8XC251SB instructions require A5 prefix
Page	Configures in Non-page mode or Page mode = 1 is a non page mode that is compatible with MCS 51 microcontrollers = 0 is a Page mode which will reduce the external code fetching by half
Rd0-Rd1	Configures RD# and PSEN# to alternate functions as shown in Table 2
XALE	Configures the ALE signal to 0 or 1 wait state = 1 maintains the ALE pulse as 0 wait state (1 oscillator clock) = 0 extends the ALE pulse from 0 to 1 wait state (3 oscillator clocks)
WSa	Extends the PSEN#/RD#/WR# signals to 0 or 1 wait state when strobing for all address regions except for address region 01:0000-01:FFFF = 1 maintains the PSEN#/RD#/WR# signals as 0 wait state (1 oscillator clock) when strobing for all address regions except for address region 01:0000-01:FFFF = 0 extends the PSEN#/RD#/WR# signals to 1 wait state (3 oscillator clocks) when strobing for all address regions except for address region 01:0000-01:FFFF
-	Reserved

Table 2. Configuration bits assignment for Rd1 and Rd0

Rd1	Rd0	RD# function	PSEN# function	Features
1	1	RD# is used to strobe for external Data memory	PSEN# is used to strobe for external Code memory	compatible with MCS ® 51 microcontrollers
1	0	RD# functions as I/O pin P3.7 only	PSEN# is used to strobe for all external memories including code and data	additional one I/O pin
0	1	RD# functions as address pin A16 only	PSEN# is used to strobe for all external memories including code and data	128K extended addressability
0	0	Reserved	Reserved	Reserved

Table 3. CONFIG1: Configuration Bytes

CONFIG1 TROM Address = 81H Reset value = 11111111B

	-	-	-	INTR	WSb	-	-	EMap
bit	7	6	5	4	3	2	1	0

Symbol	Function
EMap	Maps the upper 8K of the 16K internal code memory (FF:2000-FF:3FFF) into addresses 00:E000-00:FFFF as look-up table = 1 does not map the upper 8k = 0 maps the upper 8k
WSb	Extends the PSEN#/RD#/WR# signals to 0 or 1 wait state when strobing for address region 01:0000-01:FFFF only = 1 maintains the PSEN#/RD#/WR# signals as 0 wait state (1 oscillator clock) when strobing for address region 01:0000-01:FFFF = 0 extends the PSEN#/RD#/WR# signals to 1 wait state (3 oscillator clocks) when strobing for address region 01:0000-01:FFFF
INTR	configures interrupt frame to 2 or 4 bytes = 1 4 bytes frame pushed on stack during call to Interrupt Service Routines = 0 2 bytes frame pushed on stack during call to Interrupt Service Routines which is compatible with MCS ® 51 microcontrollers
-	Reserved

Example 1

To configure the 87C251SB (EPROM part) as

- Binary Mode compatible with 87C51FB (EPROM part)
- ALE with 1 wait state
- PSEN#/RD#/WR# with 1 wait state for all the address regions
- no page mode
- 2 bytes interrupt frame
- no internal code memory mapping

The following configuration can be used

CONFIG0 TROM Address = 80H Programmed value = 11001110B

	bit 7							bit 0
	-	-	WSa	XALE	Rd1	Rd0	Page	Src
programmed value	1	1	0	0	1	1	1	0

CONFIG1 TROM Address = 81H Programmed value = 11100111B

bit 7							bit 0
-	-	-	INTR	WSb	-	-	EMap

programmed value	1	1	1	0	0	1	1	1

Example 2

To configure the 80C251SB (ROMless part) as

- Source Mode
- 128K extended addressability for external EPROM
- ALE with 0 wait state
- PSEN#/WR# with 0 wait state for all the address regions
- page mode
- 4 bytes interrupt frame
- no internal code memory mapping

The following configuration can be used

CONFIG0 TROM Address = 80H Programmed value = 11110101B

	bit 7							bit 0
	-	-	WSa	XALE	Rd1	Rd0	Page	Src
programmed value	1	1	1	1	0	1	0	1

CONFIG1 TROM Address = 81H Programmed value = 11111111B

	bit 7							bit 0
	-	-	-	INTR	WSb	-	-	EMap
programmed value	1	1	1	1	1	1	1	1

3.0 CODE COMPATIBILITY

The 8XC251SB can be configured into either Binary mode or Source mode. When 8XC251SB is configured in Binary mode, it is binary code compatible with MCS ® 51 microcontroller. In this mode, 8XC251SB can execute 8XC51FX application codes directly without any modification or rewrite. To configure 8XC251SB in Binary Mode, the configuration bit, SRC of configuration byte CONFIG0 is set to 0.

There are a total of 255 instruction mnemonics implemented in the 256 hex codes available in the MCS ® 51 microcontroller. The reserved hex location is A5H. For 8XC251SB, A5H is used as an Escape code <ESC>, a window location to generate more hex codes for new instructions. When configured in Binary mode, all the new 8XC251SB instructions are prefixed with an A5. Therefore, the length of the hex code size for new instructions is increased by one byte compared to hex code in source mode. All the hex code for MCS 51 microcontroller instructions are identical to 8XC251SB. A few hex code examples are shown in the Example 3.

It is recommended that the 8XC251SB be configured in Binary Mode when executing existing 8XC51FX code. Users do not need to reassemble/recompile the existing 8XC51FX code since the 8XC51FX code that was assembled/compiled with the existing MCS 51 microcontroller assembler/compiler can be executed directly by the 8XC251SB in Binary Mode. Also, the 8XC251SB will perform better in Binary Mode than in Source Mode for existing 8XC51FX code. Since all of the hex code instructions are the same there is no additional prefix required as long as the new MCS 251 instructions are not used.

When configured as Source mode, all the new 8XC251SB instructions do not have the A5 prefix. On the other hand all the MCS 51 microcontroller instructions in Register addressing mode now require the prefix. Hence, the length of the hex code for MCS 51 microcontroller instructions in the Register addressing mode is increased by one byte. A few hex code examples are shown in the Example 3.

To configure 8XC251SB in Source Mode, the configuration bit, SRC of configuration byte CONFIG0 is programmed to 1. For Source Mode, either existing 8XC51FX codes or/and new 8XC251SB codes need to be assembled/compiled with new MCS 251 microcontroller assembler/compiler. The 8XC251SB will perform better in Source Mode if a majority of the code is based on new 251 instructions.

Example 3

Table 4 shows a few examples of hex code for the 8XC251SB instructions.

Table 4. Hex code examples for 8XC251SB instructions

Instruction	In Binary mode	In Source mode
INC A	04	04
INC @R1	07	A507
INC R0	08	A508
JSLE rel	A508	08
ADD A, direct	25	25
ADD A, R0	28	A528
JLE rel	A528	28

4.0 WAIT STATE GENERATION

The internal code memory of the 8XC251SB is faster than most of the external memory and peripherals devices available. When interfacing the 8XC251SB with slower devices, a wait state can be used to extend the external system bus cycle. The 8XC251SB can be configured to generate 0 or 1 wait state for ALE and PSEN#/RD#/WR# signals. With 0 wait state, the ALE, PSEN#, RD# and WR# pulse widths is 1 clock period. With 1 wait state, those pulse widths are extended to 3 clock periods.

Once the ALE signal is extended to 1 wait state, the ALE pulse is always 3 clock periods whenever there is an external access to the memory. The ALE signal is only generated when it is reading from or writing to external memory devices. When it is reading from or writing to the internal memory, there is no ALE signal. This is different from the 8XC51FX where the ALE signal is always generated even though it is executing from or accessing the internal memory. The configuration bit, XALE is used to configure the length of the ALE pulse. When XALE is set to 1, the ALE pulse width is 1 clock period. The ALE pulse width becomes 3 clock periods when the XALE bit is set to 0.

If both PSEN#/RD#/WR# and ALE signals are extended to 1 wait state at the same time. The total wait states for the 8XC251SB with this configuration become 2 wait states.

For PSEN#/RD#/WR# signals, different wait state can be programmed to separate address regions to interface with the different speed of memories and peripheral devices. One wait state can be used to interface with slower memory devices or peripherals. The configuration bits, WSa and WSb are used to configure for different wait state for separate address regions. When WSa is set to 1, the PSEN#/RD#/WR# signals are 1 clock period in 00:0000-00:FFFF, FE:0000-FE:FFFF and FF:0000-FF:FFFF address regions. These signals become 3 clock periods in those regions when WSa bit is set to 0. When the configuration bit WSb is set to 1, the PSEN#/RD#/WR# signals are 1 clock period in 01:0000-01:FFFF address region. The PSEN#/RD#/WR# signal becomes 3 clock period in that address region when WSb is set to 0.

Example 4

The 80C251SB (ROMless part) is interfacing with external peripherals that are slower than the external memory (EPROM). The address region for 00:0000-00:FFFF is assigned for internal 1K RAM and FF:0000-FF:FFFF is assigned to the external memory (EPROM). Since the access time for external memory (EPROM) is faster than the external peripherals, the address region for 00:0000-00:FFFF and FF:0000-FF:FFFF are programmed with 0 wait state. The address region 01:0000-01:FFFF is reserved for the external peripheral devices that are slower than the external memories. This address region is programmed with 1 wait state and the address for external peripheral devices is assigned to this region.

Configure WSa to 1 and WSb to 0. Program the external EPROM at address region FF:0000-FF:FFFF. Map all the external peripheral devices to address region 01:0000-01:FFFF. The address region for 00:0000-00:401F is reserved for internal 1K Bytes RAM.

5.0 EXTENDED STACK

The 8XC251SB has a 24-bit Program Counter (PC), 16-bit Stack Pointer (SP) and two Program Status Word registers, PSW and PSW1. PSW1 has some of the flags defined in the PSW, but the additional new flags are Negative (N) and Zero (Z) flags. Since the size of the PC and PSW1 are different from the 8XC51FX, the 8XC251SB must be configured to have 2 or 4 bytes frame pushed on the stack during call to Interrupt Service Routine. Configuration bit INTR of the configuration byte Config1 is assigned for this configuration.

To be compatible with 8XC51FX, INTR can be configured to 0. This is specially for users who use instruction RETi to return from both Interrupt Service Routine and Call Subroutine. Then the INTR bit must be configured to 0. For any MCS ® 51 microcontroller existing application codes and Library Routines, it is recommended to use RET in the Call Subroutines to return to the main program. RETi is only used to return from the Interrupt Service Routines. This is because the instruction RET will either pop 2 or 3 bytes from the stack depending on the Call Subroutines that is either ACALL or LCALL

When the 8XC251SB is configured to have 2 byte frame pushed, the lowest two bytes of the PC will be pushed to the stack during Interrupt Service Routines. To return from the Interrupt Service Routine, instruction RETi is used to pop two bytes from the stack.

To protect the contents of PSW1 and allow 8XC251SB to interface with 128K external memory, the INTR bit is programmed to 1. In this configuration, there are 4 bytes: 3 bytes for PC and one byte for PSW1 will be pushed on the stack during the Interrupt Service Routines. These four bytes are in this sequence: PSW1, high byte of PC, low byte of PC and middle byte of PC. To return from the Interrupt Service Routine, instruction RETi is used to pop the four bytes from the stack.

6.0 TIMING LOOP/SEQUENCE CONSIDERATION

The 8XC251SB uses the new MCS ® 251 microcontroller core architecture that is different from the traditional MCS 51 microcontroller's architecture. It is designed based on a pipelined architecture and register based machine. Therefore the execution time is different from the standard MCS 51 architecture. With the pipeline full, most of the instructions can be executed in 1 state (or 2 oscillator clocks) when running from internal code memory. It can be executed in 2 states (or 4 oscillator clocks) when running from an external code memory.

Since the 8XC51FX takes 6 states (or 12 oscillator clocks) per machine cycle to execute most of the instructions, the instruction execution time for 8XC251SB has been significantly improved. Therefore, users must consider changing the timing loop or sequence of 8XC51FX code used in the 8XC251SB application.

Example 5

A delay routine is used as a timing loop in 12 MHz 87C54 (EPROM part) application. The code is executed from internal memory. The code and time taken for some of the instructions when executing with the 87C54 are shown in Listing 1.

Listing 1. Time taken to execute a timing loop in 12MHz 87C54

```
;Case 1

;          Internal code execution          Time taken for 87C54 in number of states
           MOV     R0, #06H                 ;6
LOOP:      DJNZ    R0, LOOP                 ;12

;Total oscillator clock periods = [(6 x 12) + 6] x 2 = 156

;Total time taken in this timing loop = 156 x 83.33 ns = 13 us
```

In the same application, 87C54 is replaced by a 12 Mhz 87C251SB (EPROM part) in binary mode, 0 wait state for all address regions. The code is executed from internal memory. No modification is made to the code. The code and time taken for some of the instructions when executing with 87C251SB are shown in Listing 2.

Listing 2. Time taken to execute a timing loop in 12MHz 87C251SB

```
;Case 2
;          Internal code execution          Time taken for 87C251SB in number of states
           MOV     R0, #06H                 ;1
LOOP:      DJNZ    R0, LOOP                 ;2/5 (5 for looping and 2 for branching out from the loop)
```

```
;Total oscillator clock periods = [(5 x 5 + 2) + 1] x 2 = 56

;Total time in this timing loop  = 56 x 83.33 ns = 4.66 us
```

In Case 2, without changing the code, the time taken to execute the same timing loop is shorter than Case 1. To maintain the same delay time, some modification is needed. The code and time taken for some of the instructions when executing with 87C251SB after the modification on the code are shown in Listing 3. After the modification, the number of loops is increased from 6 to 15 times.

Listing 3. Time taken to execute a timing loop in 12MHz 87C251SB after modification

```
;Case 3
;       Internal code execution          Time taken for 87C251SB in number of states
        MOV     R0, #0FH                 ;1
LOOP:   DJNZ    R0, LOOP                 2/5 (5 for looping and 2 for branching o ut from the loop)

;Total oscillator clock periods = [(15 x 5 + 2) + 1] x 2 = 156

;Total time in this timing loop  = 156 x 83.33 ns = 13 us
```

7.0 EXTERNAL MEMORY ACCESS

The system bus cycle for the 8XC251SB is different from the 8XC51FX. The 8XC251SB needs 2 states (or 4 oscillator clock periods) to read an instruction or data from external memory. An external data writes cycle requires 3 states (or 6 oscillator clock periods. These situations are based on 0 wait state for all address regions and without page mode configuration. The 8XC51FX needs 3 states (or 6 clock periods) to read an instruction from external memory. External data read and write cycles require 6 states (or 12 oscillator clock periods. Hence, the system bus cycles for 8XC251SB are faster than 8XC51FX. Figure 2 shows the system bus timings for 8XC251SB.

To interface with external memory devices, three of the 8XC251SB AC timings should be considered. These AC timings are :-

- Address valid to valid Data/Instruction in (T_{AVDV1})
- RD#/PSEN# low to valid Data/Instruction in (T_{RLDV})
- Data/Instruction float after RD#/PSEN# high (T_{RHDZ})

For a system that has PSEN#/RD# signal connected directly to the (OE) pin of the external memory and the external memory device is already selected (example the CE# pin is hardwired to ground), the guidelines for these three AC timings are shown below :-

T_{AVDV1} >= T_{ACC} (access time of memory devices)

T_{RLDV} >= T_{OE} (output enable time of memory devices)

T_{RHDZ} >= T_{DF} (output disable to output float of memory devices)

For external code and data access, there are many new instructions that can be used since the 8XC251SB is a register based architecture. For 8XC51FX, the MOVC instruction is used to move a constant from code space and the MOVX instruction is used to read data from and write data to external Data memory. These instructions can be replaced by using the new 16 bit indirect addressing (MOV @Wrj or MOV @DRk) and 16 bit indirect displacement addressing (MOV @WRj+Disp or MOV @DRk+Dips) instructions.

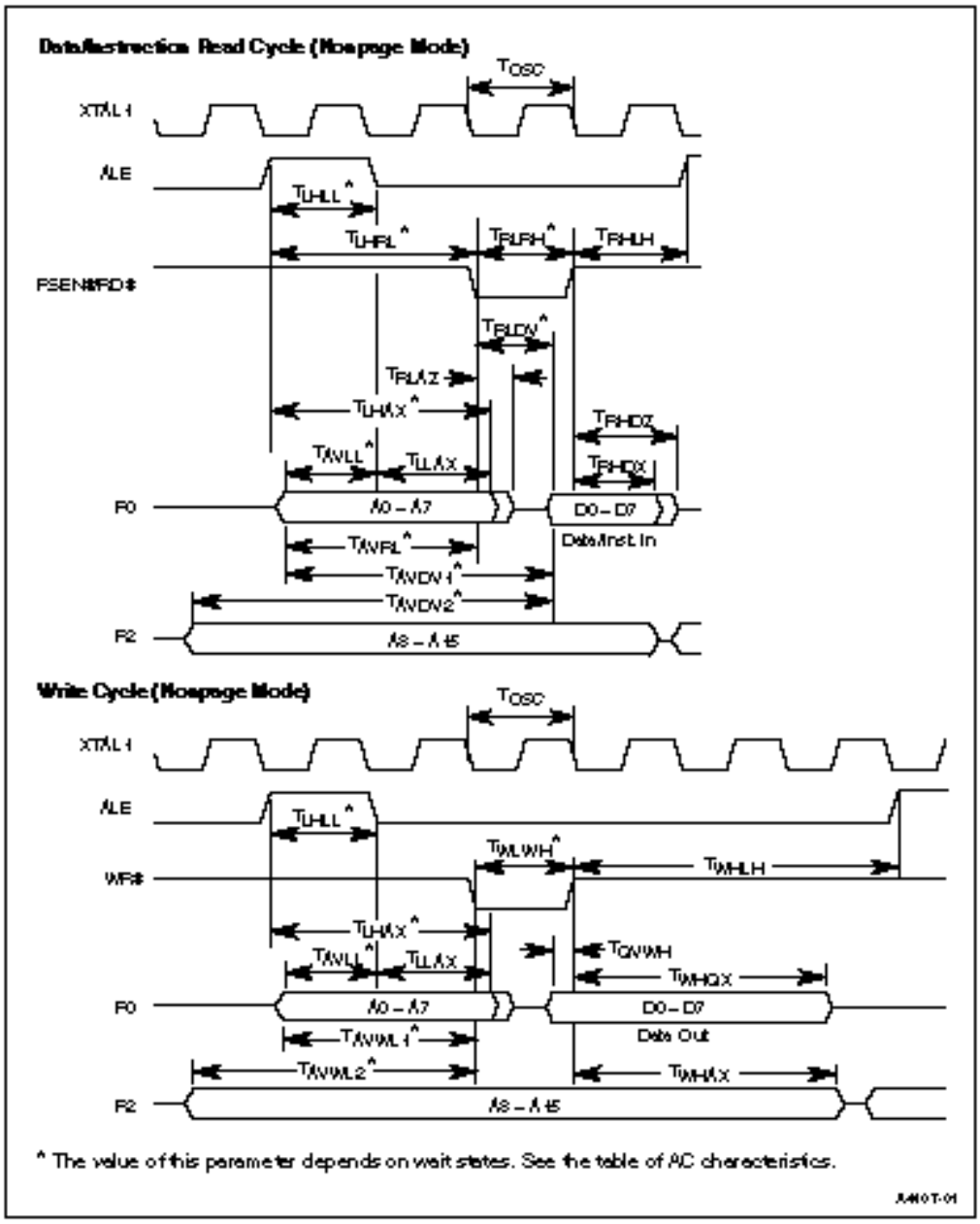

Figure 2. External Bus cycles for Data/Instruction

Read and Data write in non page mode configuration

8.0 ACCESS TIME FOR I/O PORT

The execution time for 8XC251SB to read from or write to I/O ports is different from 8XC51FX. Table 5 shows state timing for the instructions that are related to I/O ports. There are a total of 5 cases shown in the table. All the numbers shown in the table are in states where 1 state is equivalent to 2 oscillator clock periods. The 5 cases are listed below :-

Case 0 = Executing from internal EPROM with 0 wait state for ALE and PSEN#/RD#/WR# in all address regions and accessing internal RAM memory locations only.

Case 1 = Executing from internal EPROM with 0 wait state for ALE and PSEN#/RD#/WR# in all address regions and accessing I/O port SFR's

Case 2 = Executing from external EPROM with 0 wait state for ALE and PSEN#/RD#/WR# in all address regions and accessing I/O port SFR's

Case 3 = Executing from external EPROM with 1 wait state for PSEN#/RD#/WR# only in all regions and accessing I/O port SFR's

Case 4 = Executing from external EPROM with 1 wait state for PSEN#/RD#/WR# and ALE in all address regions and accessing I/O port SFR's

For Case 0, 8XC251SB can be configured in Binary Mode or Source Mode. For Case 1, 2, 3 or 4, the time taken to execute the instructions listed in the Table 5 is an adding factor to Case 0.

Table 5. State timing when accessing I/O ports of 8XC251SB

Instruction Group	Binary Case 0	Source Case 0	Add for Case 1	Add for Case 2	Add for Case 3	Add for Case 4
anl/orl/xrl/add/addc/subb A, dir8	1	1	1	2	3	4
anl/orl/xrl/add/sub/cmp Rm, dir8	3	2	1	2	3	4
anl/orl/xrl dir8, A	2	2	2	4	6	8
anl/orl/xrl dir8, #data	3	3	1	2	3	4
inc/dec dir8	2	2	2	4	6	8
anl/orl C, bit51	1	1	1	2	3	4
anl/orl C, /bit51	1	1	1	2	3	4
anl/orl C, bit	3	2	1	2	3	4
anl/orl C, /bit	3	2	1	2	3	4
setb/clr/cpl bit51	2	2	2	4	6	8
setb/clr/cpl bit	4	3	2	4	6	8
mov C, bit51	1	1	1	2	3	4
mov C, bit	3	2	1	2	3	4
mov bit51, C	2	2	2	4	6	8
mov bit, C	4	3	2	4	6	8
mov A, dir8	1	1	1	2	3	4
mov Rn, dir8	1	2	1	2	3	4
mov Rm, dir8	3	2	1	2	3	4
mov dir8, A	2	2	1	2	3	4
mov dir8, Rn	2	3	1	2	3	4
mov dir8, Rm	4	3	1	2	3	4
mov dir8, #data	3	3	1	2	3	4
xch A, dir8	3	3	2	4	6	8

Example 6

The 87C251SB (EPROM part) is programmed in Binary mode. The code is executed from internal EPROM. RAM_LOCATION is a memory location pre-defined in the internal Data RAM. PORT1 is Port 1 that is pre-defined in the SFR. Part of the instruction codes is shown in Listing 4.

Listing 4. Time taken to execute in Case 0 and Case 1

```
;          Instruction code                Execution time
CASE0:
           MOV R0, #055H                   ;2 state (4 oscillator clock periods)
           ANL R0, RAM_LOCATION            ;3 state (6 oscillator clock periods)
CASE1:
           MOV R0, #0AAH                   ;2 state (4 oscillator clock periods)
           ANL R0, PORT1                   ;4 states (8 oscillator clock periods)
             .
             .
             .
```

The time taken to execute ANL instruction in Case0 is only 3 state, but the time taken for the same ANL instruction in Case1 is 4 states.

9.0 INTERRUPT LATENCY

The interrupt request and sampling for 8XC251SB have been changed to 4 states per interrupt cycle instead of the 6 states per interrupt cycle for the 8XC51FX. Therefore, the interrupt latency for 8XC251SB will vary depending on the

configurations and the operating modes. Table 6 shows how latency will vary by various configurations and operating modes. All the numbers shown in Table 6 are in states.

Table 6. Interrupt latency for different configurations and operation modes

Int/Ext Stack	Wait State Cnfg. Max(WSa,WSb)	Stack frame INTR	Int/Ext Execution	Latency
Int	-	0	Int	17 + Longest Instruction
Int	0	0	Ext	19 + Longest Instruction
Int	1	0	Ext	21 + Longest Instruction
Int	-	1	Int	22 + Longest Instruction
Int	0	1	Ext	24 + Longest Instruction
Int	1	1	Ext	26 + Longest Instruction
Ext	0	0	Int	21 + Longest Instruction
Ext	0	0	Ext	25 + Longest Instruction
Ext	0	1	Int	30 + Longest Instruction
Ext	0	1	Ext	36 + Longest Instruction
Ext	1	0	Int	23 + Longest Instruction
Ext	1	0	Ext	29 + Longest Instruction
Ext	1	1	Int	34 + Longest Instruction
Ext	1	1	Ext	42 + Longest Instruction

The latency number listed in Table 6 should be modified when additional configurations as shown below are added :-

1. If fetching code from external memory using page mode subtract 1.
2. If internal stack, fetching code from external memory and using extended ALE add 2.
3. If external stack, fetching code from internal memory, and using extended ALE add 4 for INTR=1 and 2 for INTR=0.

4. If external stack, fetching code from external memory, and using extended ALE add 10 for INTR=1 and 6 for INTR=0.

5. If the interrupt latency is from an internal peripheral event subtract 1.

10.0 EXTENDED MEMORY SPACE

The 8XC251SB provides 256 Kbytes linear address space. The 256 Kbytes linear address spaces are at regions 00:0000-00:FFFF, 01:0000-01:FFFF, FE:0000-FE:FFFF and FF:0000-FF:FFFF. At any one time, the maximum linear address space available for external memory is either 64K or 128K Bytes. The 128K Bytes address space can be used for code or data. To interface 8XC251SB with 128K external memory devices, the configuration bits Rd1 and Rd0 of configuration byte CONFIG0 must be programmed to 0 and 1 respectively. In this mode the RD# pin function as address bit A16. All the external code or data memory spaces are strobed by PSEN#.

In this mode, out of the 24-bit Program Counter (PC), 17-bit will be used for extended address. The full address range is 128K. To jump beyond 64K address region, EJMP, ECALL and ERET must be used. However, it is recommended not to use EJMP on the first line of the code if the INT0 is used at the same time. This is because the EJMP will take 4 bytes and the Interrupt vector for INT0 is at address 0003. The address for the last byte of the EJMP instruction would overlap with the INT0 Interrupt vector. The code space can be internal or external. Example 7 explains how the 8XC251SB can be used to interface with 128K external SRAM.

Example 7

87C251SB (EPROM part) has 16K of code which is stored internally. It is used to interface with 128K external SRAM. Figure 3 shows the hardware design of this memory system. Figure 4 shows the address space for this system. In this design, the lower 1K bytes of external RAM is not accessible by the user. This is because of the space is overlapping with the internal 1K bytes RAM.

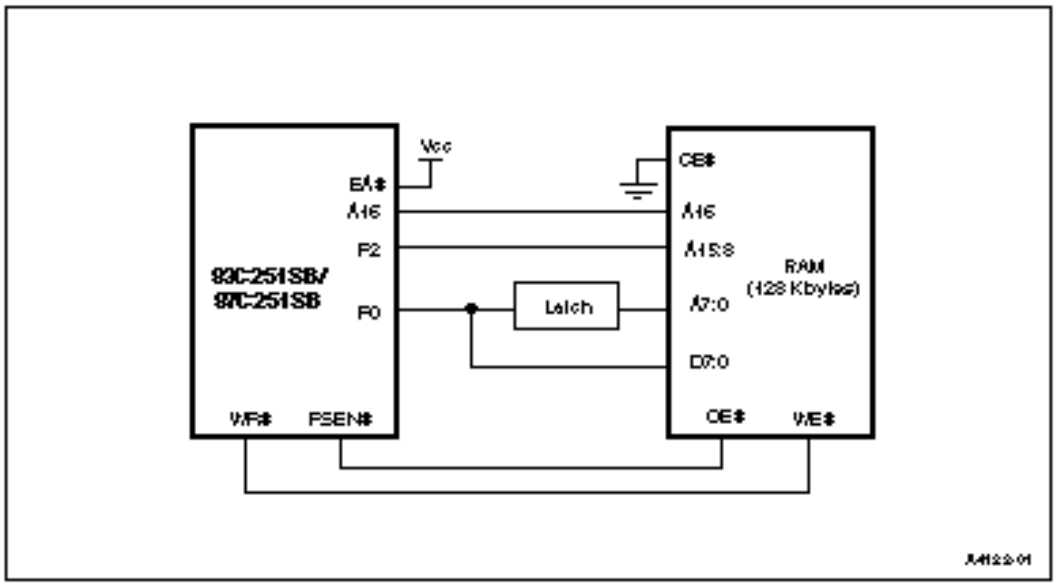

Figure 3. The 8XC251SB interfaces with 128K external SRAM

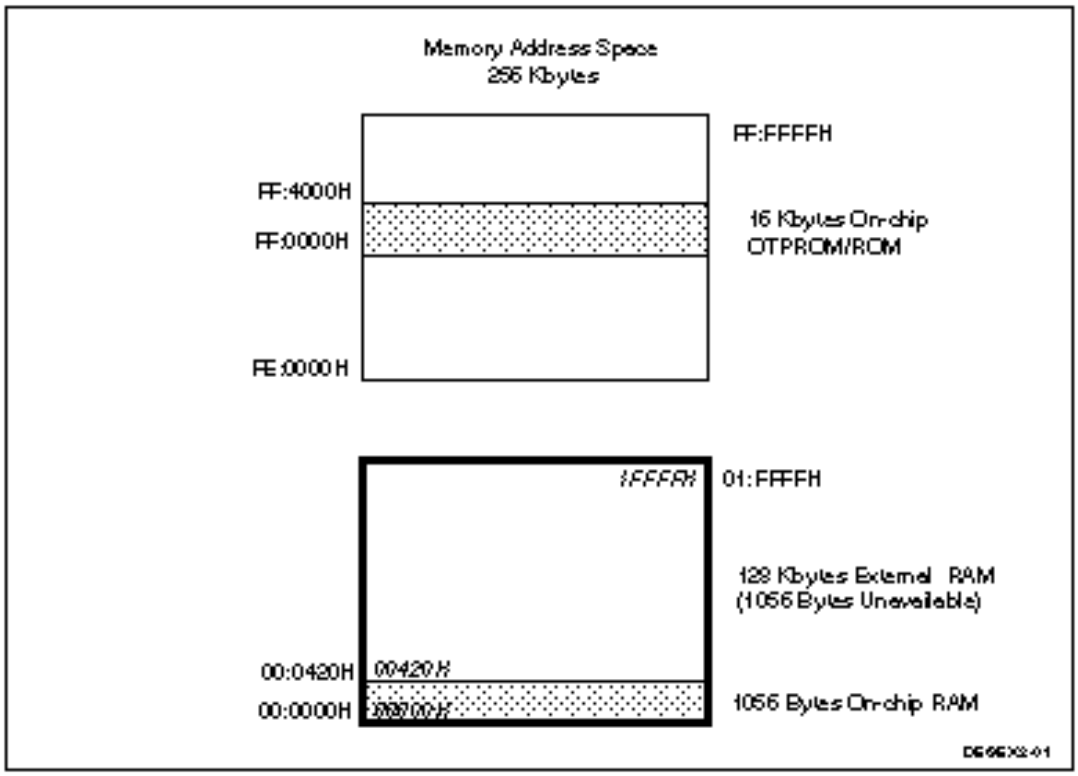

Figure 4. The address space from the system of Figure 3

11.0 PAGE MODE DESIGN

In normal operation, the 8XC251SB requires 2 states to fetch an instruction from external memory. Page mode can be used to speed-up the code fetching from the external memory. In page mode, data is multiplexed with the high byte of address (A8-A15) and is generated from Port 2. During page mode operation, 8XC251SB requires only 1 state to fetch an instruction in the page hit situation. A page consists of 256 bytes. A page hit occurs starting from the second byte until the last byte of a 256 byte page. The ALE signal is only generated in the first byte of 256 byte page. The high byte of the address (A8-A15) is held by an external latch for 256 byte of code fetching.

The page miss occurs in the first byte. It requires 2 states to fetch the first code. The page miss situations are listed as below :-

1) if the code execution branches out from this 256 byte page, a subroutine or an interrupt routine is called

2) fetching the first byte of the 256 byte page

3) fetching the first byte after the power down and idle mode

4) fetching the first byte after a page rollover

Figure 5 shows the Port 0 and 2 connection with external memory in a non page mode and page mode design. Figure 6 shows the external code bus cycle in page mode configuration. Once the 8XC251SB is configured in page mode, it is not pin compatible with 8XC51FX.

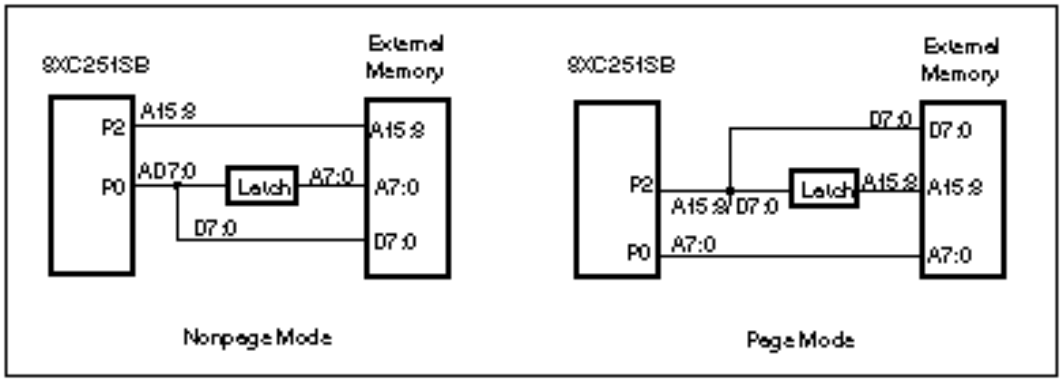

Figure 5. Port 0 and 2 connection with external memory in non page mode and page mode

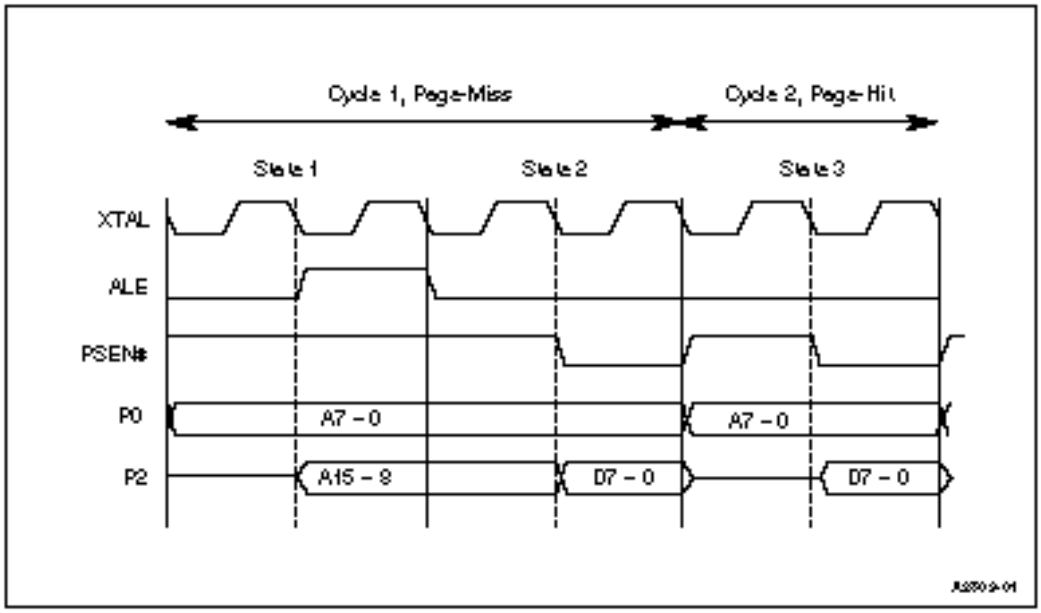

Figure 6. External Code Fetch cycle in Page Mode

12.0 ADDITIONAL REFERENCES

Literature	Order Number
1. MCS 51 Microcontroller Family User's Manual	272383
2. 8XC251SB User's Manual	272617
3. 8XC251SB High Performance CHMOS Single-Chip Microcontroller Datasheet	272616
4. Introducing The MCS 251 Microcontroller - 8XC251SB (Application Note AP-708)	272670
5. Maximizing Performance Using MCS 251 Microcontroller - Programming 8XC251SB (Application Note AP-709)	272671